Math 421 • November 2009

Complex *n*th roots

Prerequisites

Mathematica

Most of this notebook requires David Park" *Mathematica* add-on application *Presentations*. If this is not already available to you, see Park's Mathematica page for how to obtain it.

In order to reproduce all the graphics output from this notebook, you must have installed that application and then initialized it by evaluating, for example, the expression:

```
Needs["Presentations`Master`"]
```

That initialization is done below when it is first needed.

Mathematics

You should already know the basics of the algebra of complex numbers—how to add and multiply them—and the representation of complex numbers by points in the plane.

The *n*th roots of a complex number

For a positive integer n=1, 2, 3, …, a complex number $w \neq 0$ has n different complex roots z. That is, for a given $w \neq 0$, the equation $z^n = w$ has n different solutions z.

This is the case, in particular, when $w = 1$. In this case, the n different values of z are called **the *n*th roots of unity**.

This notebook shows how to use *Mathematica* to calculate such roots as well as how to visualize them geometrically. It also includes material about expressing complex roots of unity in "polar form".

The cube roots of unity

For an example, work with the cube roots of unity. By definition, a **cube root of unity** is a solution of the equation

$$z^3 = 1.$$

Surely *Mathematica* can solve this equation directly. Try it:

In[1]:=
```
Solve[z^3 == 1, z]
```

Out[1]=
$$\{\{z \to 1\}, \{z \to -(-1)^{1/3}\}, \{z \to (-1)^{2/3}\}\}$$

That's not useful! What are the solutions, really? Use `ComplexExpand` on the result of `Solve`:

In[2]:=
```
cubeRootsUnity = ComplexExpand[z /. Solve[z^3 == 1, z]]
```

Out[2]=
$$\left\{1, -\frac{1}{2} - \frac{i\sqrt{3}}{2}, -\frac{1}{2} + \frac{i\sqrt{3}}{2}\right\}$$

How can you be sure that there are no more than just these three roots? This is a consequence of the Fundamental Theorem of Algebra. (See the notebook FactorTheorem.nb.)

Visualizing the cube roots of unity

The graphics functions introduced below are used here just for visualizing the cube roots of unity. But they can be used more generally to visualize diverse sets of complex numbers.

You'll need a number of computational and graphics functions that are not already built-in objects in *Mathematica*. They are defined in David Park's *Presentations* application. Tell *Mathematica* you are going to use that application:

In[3]:=
```
Needs["Presentations`Master`"]
```

Plotting roots of unity as points in the plane

You'll need to convert each of the complex numbers that are the cube roots of unity into an (x, y)-coordinate pair. And then you'll need to surround it with the graphics primitive `Point` to produce a graphics object capable of being displayed. Park's function **`ComplexPoint`** does both of those things at once:

In[4]:=
```
cubeRootsUnity[[3]]
```

Out[4]=
$$-\frac{1}{2}+\frac{i\sqrt{3}}{2}$$

In[5]:=
```
ComplexPoint[cubeRootsUnity[[3]]]
```

Out[5]=
$$\text{Point}\left[\left\{-\frac{1}{2},\ \frac{\sqrt{3}}{2}\right\}\right]$$

You could do the same thing to all three roots by forming:

In[6]:=
```
Table[ComplexPoint[cubeRootsUnity[[k]]], {k, 1, 3}]
```

Out[6]=
$$\left\{\text{Point}[\{1, 0\}], \text{Point}\left[\left\{-\frac{1}{2}, -\frac{\sqrt{3}}{2}\right\}\right], \text{Point}\left[\left\{-\frac{1}{2}, \frac{\sqrt{3}}{2}\right\}\right]\right\}$$

A more direct, "functional" way to accomplish the same thing is:

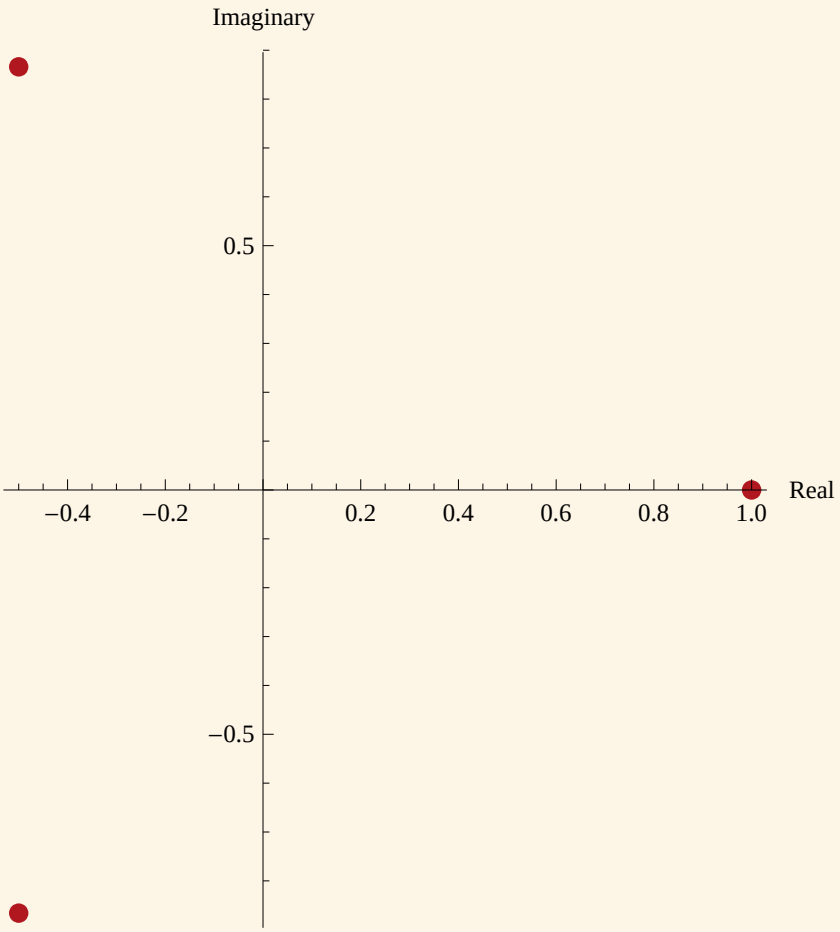
In[7]:=
```
Map[ComplexPoint, cubeRootsUnity]
```

Out[7]=
$$\left\{\text{Point}[\{1, 0\}], \text{Point}\left[\left\{-\frac{1}{2}, -\frac{\sqrt{3}}{2}\right\}\right], \text{Point}\left[\left\{-\frac{1}{2}, \frac{\sqrt{3}}{2}\right\}\right]\right\}$$

Here is the plot of the three cube roots of unity in the complex plane:

In[8]:=
```
Draw2D[
 {(* Set color and point size for plotting points *)
  Legacy@IndianRed, PointSize[Large],
  (* Draw points *)
  Map[ComplexPoint, cubeRootsUnity]
 },
 Axes → True, AxesLabel → {"Real", "Imaginary"}]
```

Out[8]=

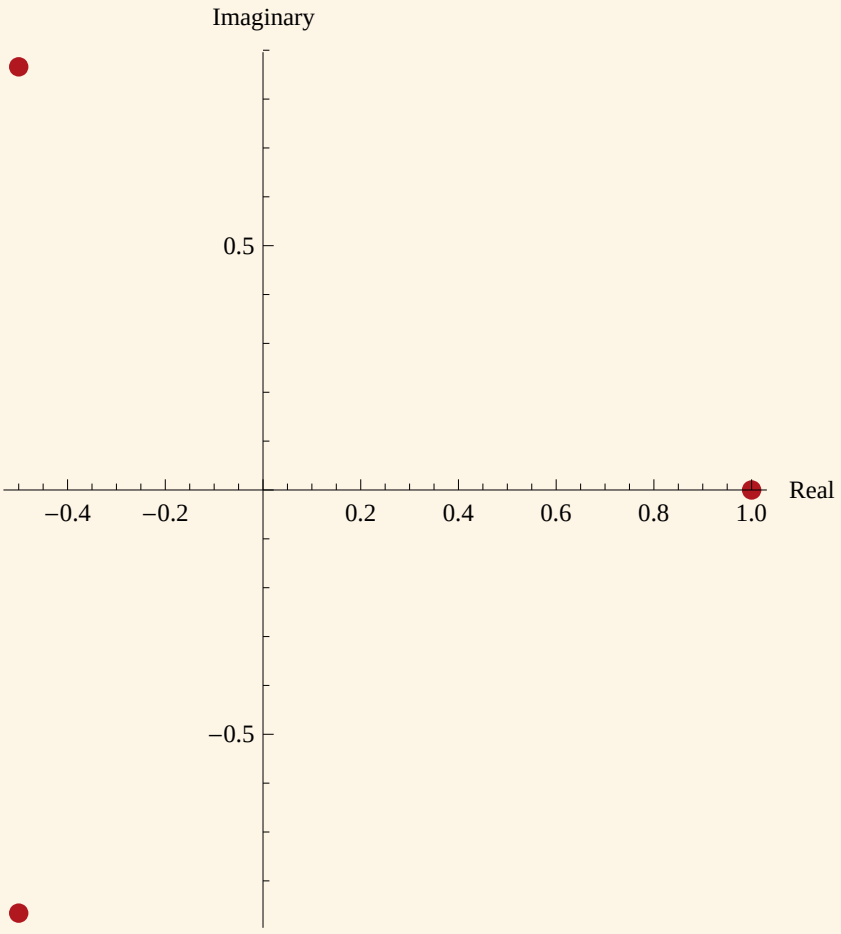


The preceding input cell included **comments**, enclosed in `(* …*)` pairs, just to help you understand what the various expressions do that constitute the entries in the (list) argument to `Draw2D`. Ordinarily such comments are not needed. (*Mathematica* documentation is more usually in text cells.)

But it does help to break the entire expression into separate lines to show successive **graphics directives**—color and point size specifications, for example—and **graphics primitives**—`ComplexPoints`, etc.—to which those directives apply. Here's the same expression as before, but with the comments omitted:

In[9]:=
```
Draw2D[
 {
  Legacy@IndianRed, PointSize[Large],
  Map[ComplexPoint, cubeRootsUnity]
 },
 Axes → True, AxesLabel → {"Real", "Imaginary"}]
```

Out[9]=



Notice that the entire argument to the function `Draw2D` is a list (with a `{}` pair).

Other named alternatives for the `PointSize` directive are `Medium`, `Small`, and `Tiny`. For more precise control, you can specify the point size by a number, which will be the number of printer's points.

Exercise. Experiment with the effects of the following:

- Changing the argument to `PointSize`
- Changing, the color from `Legacy@IndianRed` to something else

pretty.

Note: The named colors

`Red`, `Blue`, `Green`, `Orange`, `Yellow`, `Purple`, `Brown`, `Cyan`, `Magenta`, `Pink`

are built-in and immediately available.

Additional named colors are available. Use the Color Schemes palette to find them. The Named colors in the Legacy group are the ones you may access easily using the *Presentations* function `Legacy` , as with the legacy named color IndianRed used above.

Still more colors are available using directly the built-in *Mathematica* `ColorData` function.

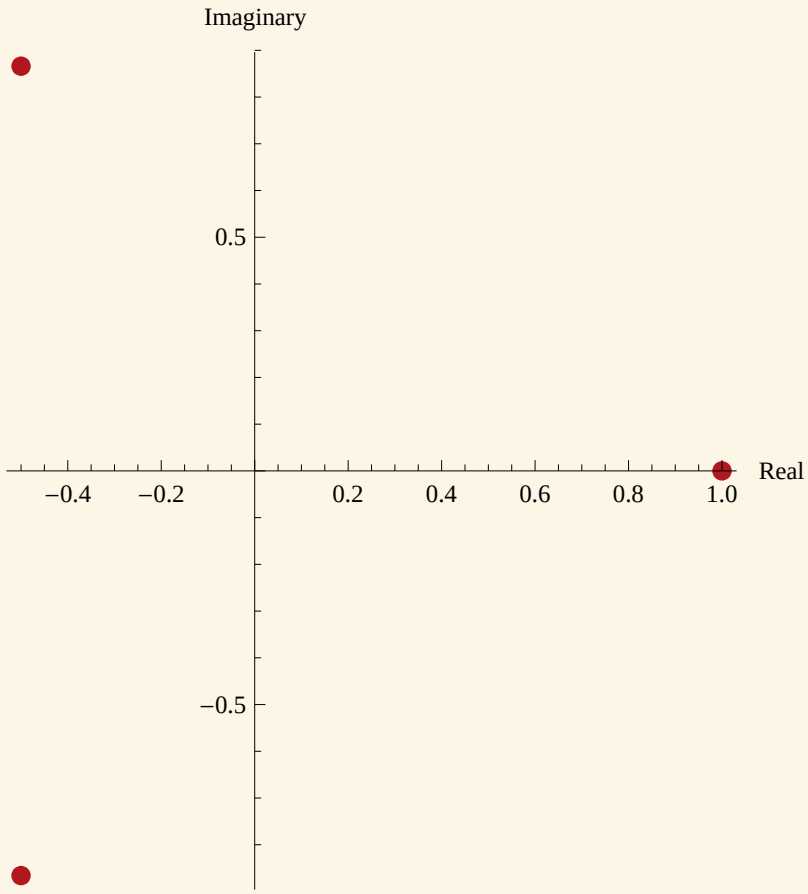
To change the overall size of the plot, include as a second argument to `Draw2D` an option of the form

`ImageSize→72` *inches*

where *inches* is the number of inches you want for the width of the plot. (*Mathematica* measures the width in "printer's points". There are 72 printer's points to the inch.)

In[10]:=
```
Draw2D[
 {
  Legacy@IndianRed, PointSize[Large],
  Map[ComplexPoint, cubeRootsUnity]
 },
 Axes → True, AxesLabel → {"Real", "Imaginary"}, ImageSize → 4 × 72]
```

Out[10]=



In[11]:=

To annotate the display, you may include text at any locations by invoking the *Presentations* function **`ComplexText`**. This function is used in the form

`ComplexText[`*txt*`,`*z*`]`

where the *txt* is the text you want to display and *z* is the complex number in the plane where (the center of) the text is to be displayed.

For example, create a label for the root $1 = 1 + 0\,i$ of unity like this…

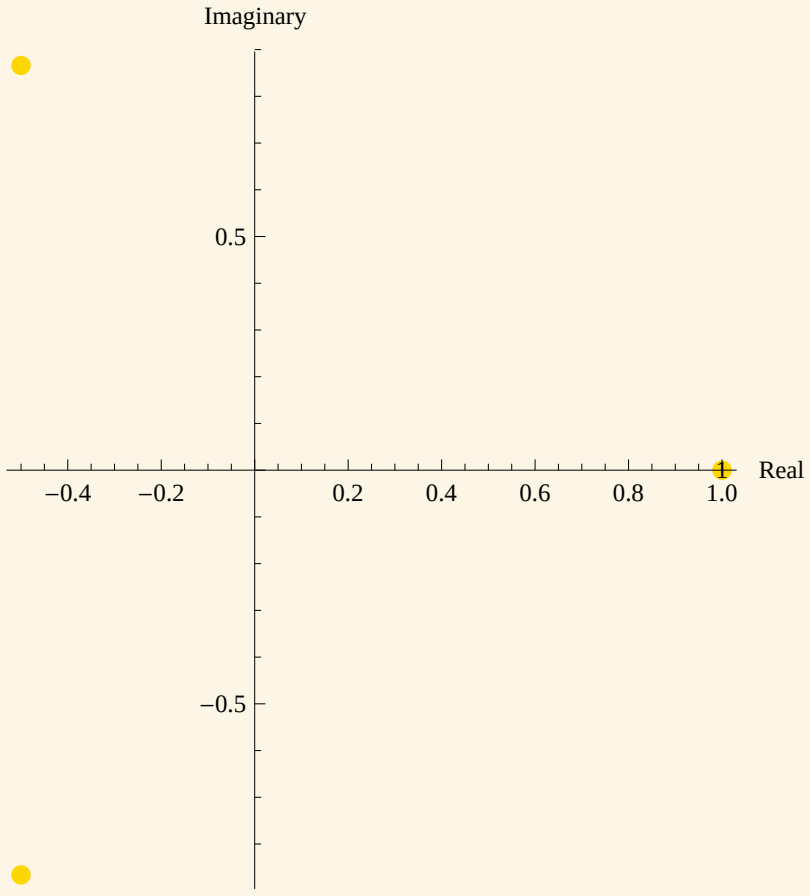
In[12]:=
```
ComplexText[TraditionalForm[cubeRootsUnity[[1]]], cubeRootsUnity[[1]]]
```

Out[12]=
```
Text[1, {1, 0}]
```

…and include that label like this:

In[13]:=
```
Draw2D[
 {
  Legacy@Gold, PointSize[Large],
  Map[ComplexPoint, cubeRootsUnity],
  Black, ComplexText[
   TraditionalForm[cubeRootsUnity[[1]]], cubeRootsUnity[[1]]]
 },
 Axes → True, AxesLabel → {"Real", "Imaginary"}, ImageSize → 4 × 72]
```

Out[13]=



The point color there was changed to `Gold` so that you can see that the label was displayed directly over the point. Ordinarily you will want to use some "offset" to

move the text away from the point…

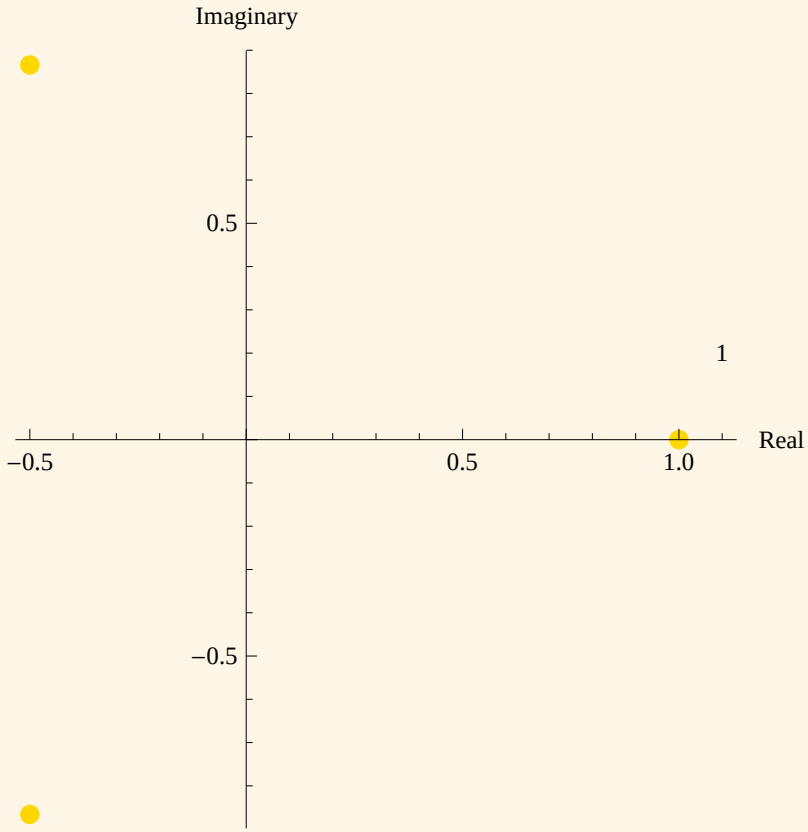
In[14]:=
```
incr = 0.1 + 0.2 i;
ComplexText[TraditionalForm[cubeRootsUnity[[1]]],
 cubeRootsUnity[[1]] + incr]
```

Out[15]=
```
Text[1, {1.1, 0.2}]
```

…and include that in the `ComplexText` item of the `Draw2D` expression:

In[16]:=
```
Draw2D[
 {
  Legacy@Gold, PointSize[Large],
  Map[ComplexPoint, cubeRootsUnity],
  Black, ComplexText[
   TraditionalForm[cubeRootsUnity[[1]]], cubeRootsUnity[[1]] + incr]
 },
 Axes → True, AxesLabel → {"Real", "Imaginary"}, ImageSize → 4 × 72]
```

Out[16]=



You may form labels for all three points at once like this:

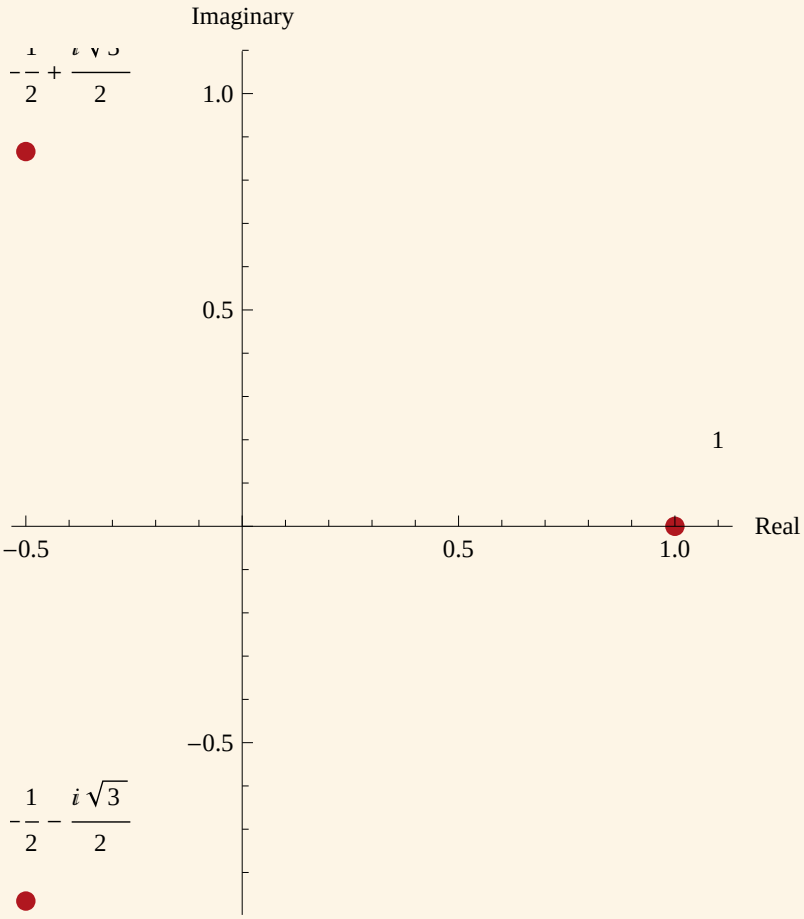
In[17]:=
```
Map[ComplexText[TraditionalForm[#], # + incr] &, cubeRootsUnity]
```

Out[17]=
$$\left\{\text{Text}[1, \{1.1, 0.2\}], \text{Text}\left[-\frac{1}{2} - \frac{i\sqrt{3}}{2}, \{-0.4, -0.666025\}\right], \text{Text}\left[-\frac{1}{2} + \frac{i\sqrt{3}}{2}, \{-0.4, 1.06603\}\right]\right\}$$

Then here's the display of the three cube roots of unity with each labeled with its value (and the point color restored to `IndianRed`):

In[18]:=
```
Draw2D[
 {
  Legacy@IndianRed, PointSize[Large],
  Map[ComplexPoint, cubeRootsUnity],
  Black,
  Map[ComplexText[TraditionalForm[#], # + incr] &, cubeRootsUnity]
 },
 Axes → True, AxesLabel → {"Real", "Imaginary"}, ImageSize → 4 × 72]
```

Out[18]=



*Oops! O*ne of the text labels is clipped at its top, and two run into the left edge of the frame. What's more, the whole plot has the origin off-center. To fix all this at the same time, include the **PlotRange** option to `Draw2D` . The value of `PlotRange` specifies the graphics "window" (just like for a calculator plot) and is used in the form:

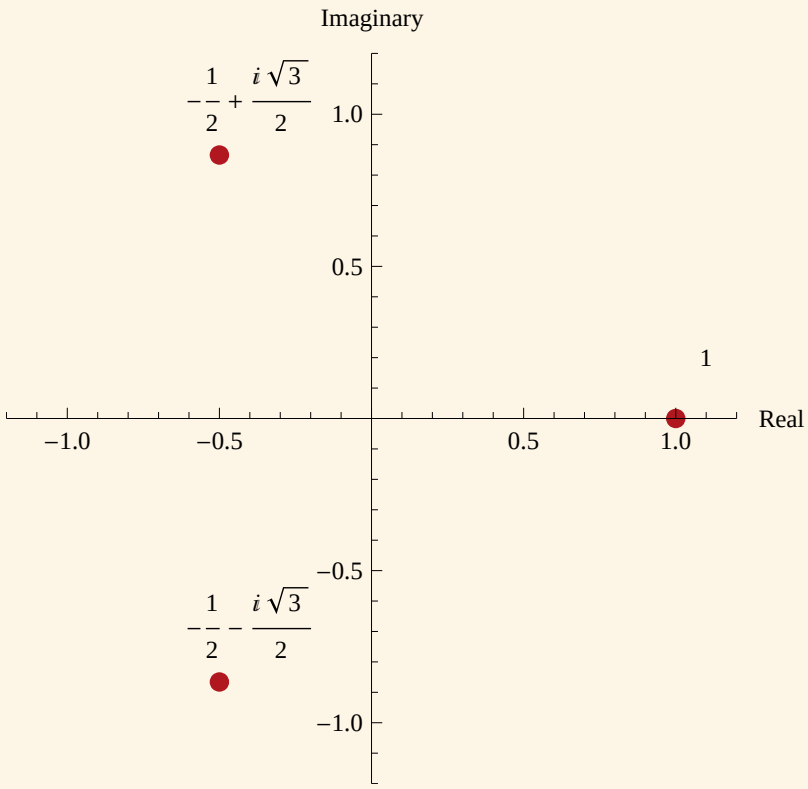
$$\{\{xmin, xmax\}, \{ymin, ymax\}\}$$

You'll often need to do some figuring and some experimenting to adjust the value

for `PlotRange` so as to provide appropriate dimensions. (Good graphics takes work—no matter how good the software creating it!)

In[19]:=
```
Draw2D[
 {
  Legacy@IndianRed, PointSize[Large],
  Map[ComplexPoint, cubeRootsUnity],
  Black,
  Map[ComplexText[TraditionalForm[#], # + 0.1 + 0.2 ⅈ] &,
   cubeRootsUnity]
 },
 PlotRange → {{-1.2, 1.2}, {-1.2, 1.2}},
 Axes → True, AxesLabel → {"Real", "Imaginary"}, ImageSize → 4 × 72]
```

Out[19]=

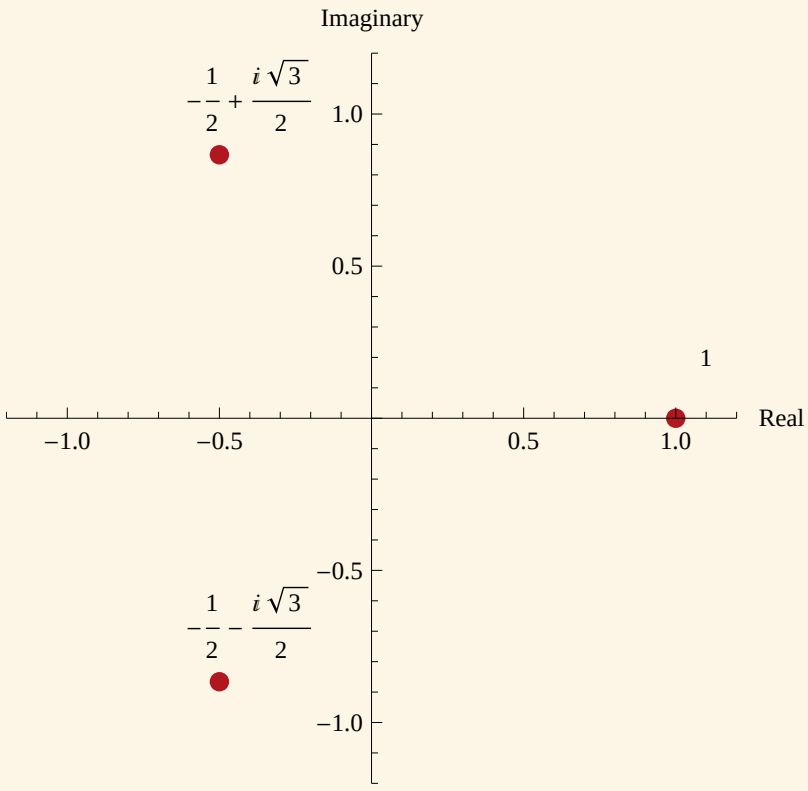


When the entire `Draw2D` expression such as the preceding one gets too long for your comfort, you may want to define first the graphics objects and then reference those names within the `Draw2D` expression. For example:

In[20]:=
```
points = Map[ComplexPoint, cubeRootsUnity];
incr = 0.1 + 0.2 I;
labels =
  Map[ComplexText[TraditionalForm[#], # + incr] &, cubeRootsUnity];

Draw2D[
 {
  Legacy@IndianRed, PointSize[Large],
  points, Black, labels
 },
 PlotRange → {{-1.2, 1.2}, {-1.2, 1.2}},
 Axes → True, AxesLabel → {"Real", "Imaginary"}, ImageSize → 4 × 72]
```

Out[23]=



Plotting roots of unity as vertices of a triangle

The relevant graphics primitive from *Presentations* here is **ComplexLine**. An expression of the form

```
ComplexLine[{z1,z2}]
```

represents the line segment from the point *z1* to the point *z2*. More generally, an expression of the form

```
ComplexLine[{z1, z2, …, zn}]
```

represents the "broken line"—the polygonal curve—with vertices *z1, z2, …, zn*. For example:

In[24]:=
```
ComplexLine[cubeRootsUnity]
```

Out[24]=
$$\text{Line}\left[\left\{\{1, 0\}, \left\{-\frac{1}{2}, -\frac{\sqrt{3}}{2}\right\}, \left\{-\frac{1}{2}, \frac{\sqrt{3}}{2}\right\}\right\}\right]$$

The result of `ComplexLine[{z1, z2, …, zn}]` is the same as that of

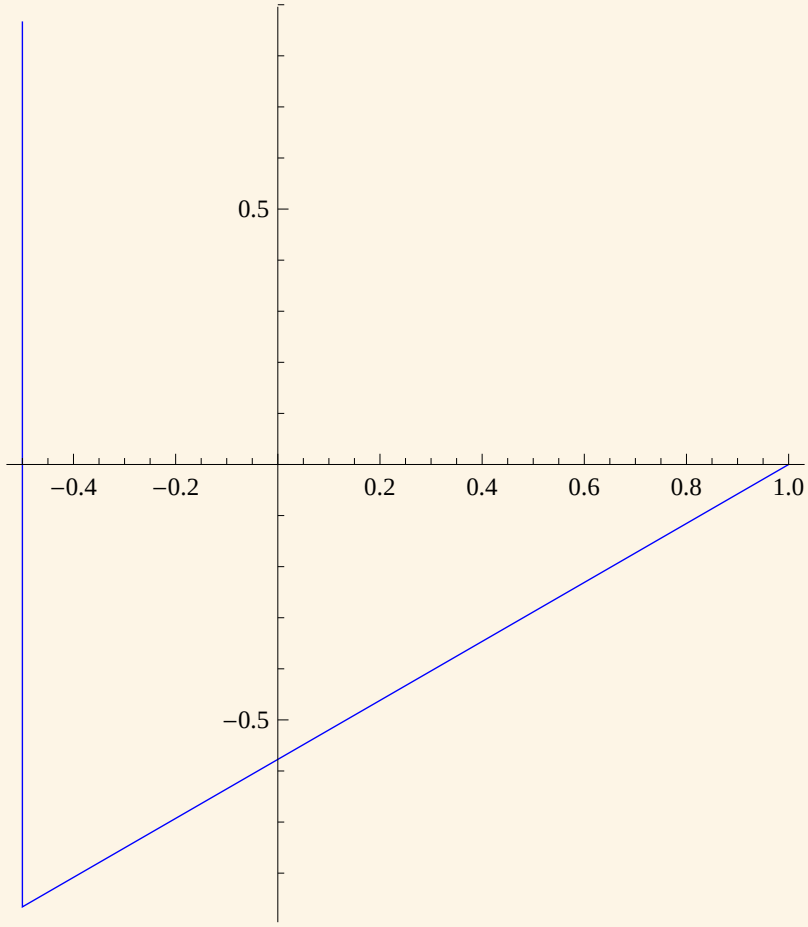
```
Line[{ToCoordinates[z1], ToCoordinates[z2], …, ToCoordinates[zn]}]
```

where the Presentations function ToCoordinates converts a complex number to the corresponding list of its real and imaginary parts. The function `Line` is a built-in *Mathematica* graphics primitive.

Here is the plot of that polygonal line:

In[25]:=
```
Draw2D[
 {
  Blue, ComplexLine[cubeRootsUnity]
 },
 Axes → True, ImageSize → 4 × 72]
```

Out[25]=



To close up such a polygonal curve to form a polygon, you need to append the first point to the end of the list:

In[26]:=
```
Append[cubeRootsUnity, First[cubeRootsUnity]]
```

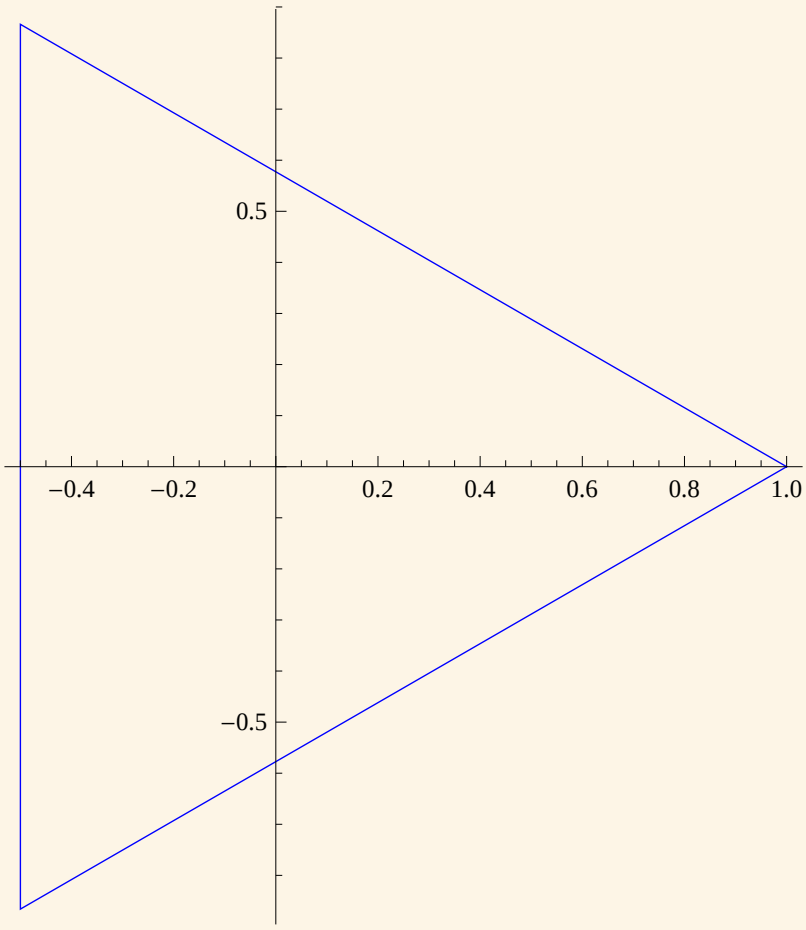
Out[26]=
$$\left\{1, -\frac{1}{2} - \frac{i\sqrt{3}}{2}, -\frac{1}{2} + \frac{i\sqrt{3}}{2}, 1\right\}$$

In[27]:=
```
ComplexLine[Append[cubeRootsUnity, First[cubeRootsUnity]]]
```

Out[27]=
$$\text{Line}\left[\left\{\{1, 0\}, \left\{-\frac{1}{2}, -\frac{\sqrt{3}}{2}\right\}, \left\{-\frac{1}{2}, \frac{\sqrt{3}}{2}\right\}, \{1, 0\}\right\}\right]$$

In[28]:=
```
Draw2D[
 {
  Blue, ComplexLine[Append[cubeRootsUnity, First[cubeRootsUnity]]]
 },
 Axes → True, ImageSize → 4 × 72]
```

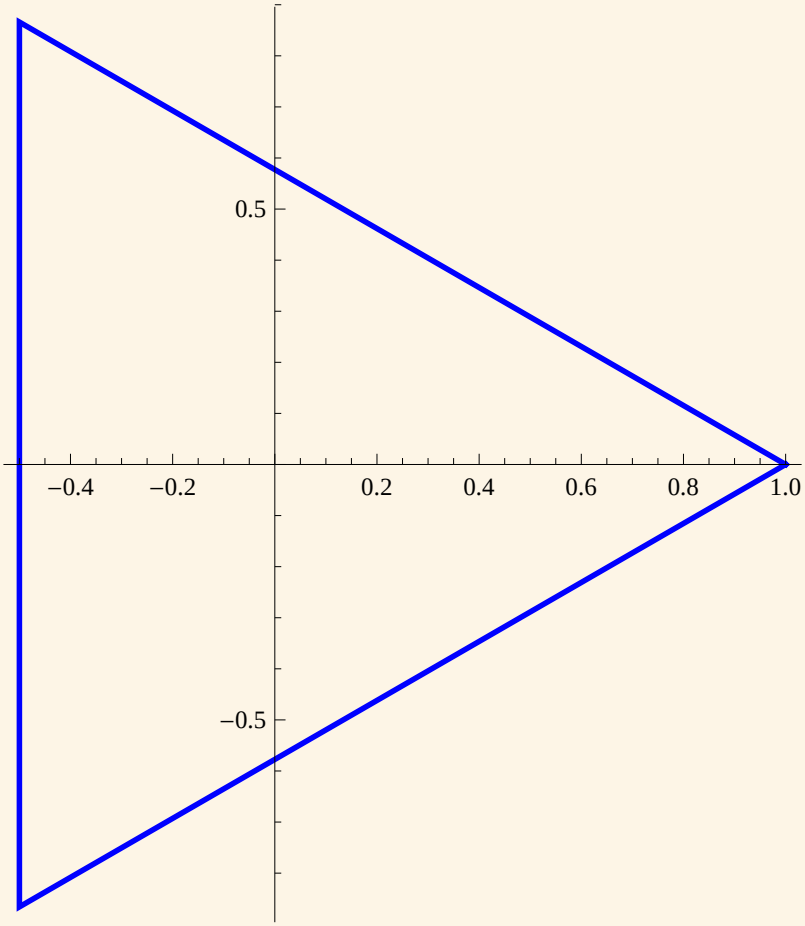
Out[28]=



To thicken the line segments in the triangle so as to make them more visible, include the `Thick` directive (or a `Thickness`[*size*] directive) before the `ComplexLine` object, like this:

In[29]:=
```
Draw2D[
 {
  Blue, Thick,
  ComplexLine[Append[cubeRootsUnity, First[cubeRootsUnity]]]
 },
 Axes → True, ImageSize → 4 × 72]
```

Out[29]=



Exercise. Display the vertices of this triangle, as enlarged points, along with the triangle itself on a single `Draw2D` display. Use different colors for the triangle and its vertices.

Exercise. Include in the display text labels for the three points.

Be sure to do the following mathematical exercise now.

Exercise. What kind of triangle is it that has the cube roots of unity as its ver-

tices? Base your answer upon what you see in the graphics display. Be as specific as you.

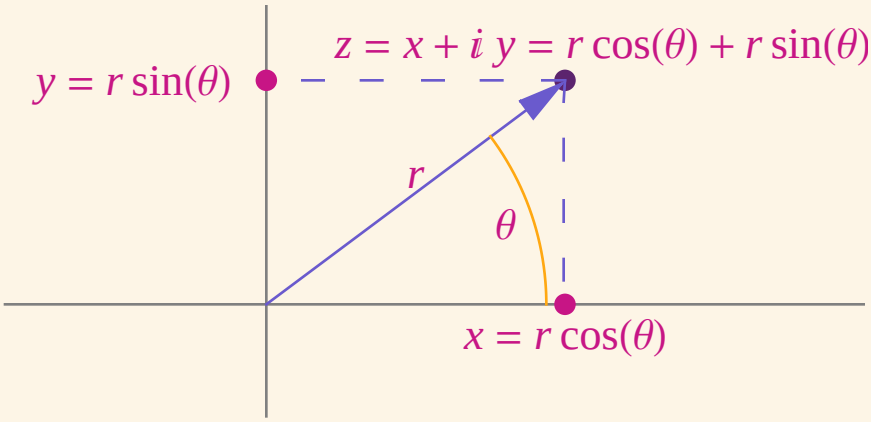
Interlude: the principal argument of a complex number

Polar representations and arguments of a complex number

A complex number $z = a + b\,i$ is represented by the point (a, b) in the plane. The numbers a and b are the usual Cartesian coordinates of that point. Such a point can also be described by polar coordinates r and θ with $a = r\cos\theta$ and $b = r\sin\theta$. In other words, the complex number z has a **polar representation**

$$z = r\cos\theta + i\,r\sin\theta$$

with $r \geq 0$. (The number $r = 0$ if and only if $z = 0$.) See the figure below.



The preceding figure was created in my notebook `PrepRolarFigure.nb` and copied to the preceding cell.

For example, the third cube root of unity ω_3 found above was:

In[30]:=
```
ω3 = cubeRootsUnity[[3]]
```

Out[30]=
$$-\frac{1}{2} + \frac{i\sqrt{3}}{2}$$

(The preceding input cell used a suffix 3 to form the name $\omega 3$ rather than use a subscript 3 to form the name ω_3, because employing subscripts in *Mathematica* names presents certain technical difficulties.)

Since $\cos 2\pi/3 = -1/2$ and $\sin 2\pi/3 = \sqrt{3}/2$, this cube root of unity ω_3 has the following polar representation:

```
In[31]:= Cos[2/3 π] + i Sin[2/3 π]
```

$$\text{Out[31]=}\quad -\frac{1}{2} + \frac{i\sqrt{3}}{2}$$

And that means that $\theta = 2\pi/3$ is an argument of the cube root of unity ω_3.

Exercise. Find two other arguments of ω_3, one positive and the other negative.

Exercise. Find an argument for each of the other two cube roots of unity.

Exercise. An earlier exercise asked you to say, from what you saw in the graphics display, what kind of triangle it is that has the cube roots of unity as its vertices. Use the results already obtained in this section to confirm mathematically what you said.

The principal argument of a complex number

When $z = r\cos\theta + i\, r\sin\theta$ with $r > 0$, necessarily,

$$r = |z|,$$

as the following *Mathematica* computation indicates.

In[32]:=
```
Clear[z, r, θ]
z = r Cos[θ] + i r Sin[θ]
Simplify[ComplexExpand[Abs[z]], r > 0 ]
Clear[z, r, θ]
```

Out[33]= `r Cos[θ] + i r Sin[θ]`

Out[34]= `r`

Because cos and sin are periodic, a single point (a, b) in the plane has infinitely many different polar coordinates. But when $(a, b) \neq (0, 0)$, that is, when the complex number $z = a + b\,i \neq 0$, then the point has *unique* polar coordinates r and θ for which

$$z = r\cos\theta + i\,r\sin\theta, \qquad r > 0, \qquad -\pi < \theta \leq \pi.$$

This unique θ in the half-open, half-closed interval $(-\pi, \pi]$ is called **the principal argument** of z.

For example, the principal argument of $\omega_3 = -1/2 + i\sqrt{3}/2$ is the number $\theta = 2\pi/3$.

⚠ You may be accustomed to selecting polar coordinates θ with $0 \leq \theta < 2\pi$. However, in complex analysis $-\pi < \theta \leq \pi$ is the convention usually adopted for the principal argument.

Exercise. Check that *neither* of the other two arguments θ of ω_3 that you found in the preceding exercise satisfies the condition $-\pi < \theta \leq \pi$ required to be the principal argument.

The *Mathematica* function `Arg` finds the principal argument. For example:

In[36]:=
```
Arg[ω3]
```

Out[36]= $\frac{2\pi}{3}$

Exercise. Find the principal arguments of the other two of the three cube roots of

unity.

Please do not proceed to the next section until you have done the preceding exercise!

The primitive cube root of unity

The principal arguments of the three cube roots of unity are:

In[37]:=
```
Arg[cubeRootsUnity]
```

Out[37]=
$$\left\{0, -\frac{2\pi}{3}, \frac{2\pi}{3}\right\}$$

The smallest strictly positive one of these three arguments is the third one, $2\pi/3$. And that is the principal argument of

$$\omega_3 = \cos 2\pi/3 + i \sin 2\pi/3.$$

This root with the smallest strictly positive principal argument is called the **primitive** cube root of unity.

According to the theory of *n*th roots of unity, the set of all three cube roots of unity consists of the powers of the primitive cube roots of unity. Here are those three powers of ω_3:

In[38]:=
```
cubeRoots = Expand[ω3^{0, 1, 2}]
```

Out[38]=
$$\left\{1, -\frac{1}{2} + \frac{i\sqrt{3}}{2}, -\frac{1}{2} - \frac{i\sqrt{3}}{2}\right\}$$

Exercise. Verify that the set of three cube roots of unity appearing in the list `cubeRootsUnity` is exactly the same as the set of powers of the primitive cube root of unity appearing in the list `cubeRoots`. Do this by evaluating a single equation in *Mathematica* that has result `True`. (*Hint:* The difficulty is that *Mathematica* lists give their entries in some particular order, whereas a set need not have any particular order for its members. Try the *Mathematica* function `Sort`.)

Roots of other complex numbers

Example: Find the cube roots of $1 + i$.

One way to find these cube roots is simply to solve the corresponding cubic equation $z^3 == 2 + 2\,i$ and, as usual, using `ComplexExpand` so as to obtain actual complex numbers in $x + i\,y$ form:

```
In[39]:= ComplexExpand[z /. Solve[z^3 == 2 + 2 i, z]]
```

$$\text{Out[39]= } \left\{\frac{1}{2} - \frac{\sqrt{3}}{2} + i\left(-\frac{1}{2} - \frac{\sqrt{3}}{2}\right),\ \frac{1}{2} + \frac{\sqrt{3}}{2} + i\left(-\frac{1}{2} + \frac{\sqrt{3}}{2}\right),\ -1 + i\right\}$$

According to the theory, you only need one of the three cube roots of $2 + 2\,i$ in order to find them all. Here's the last of the three:

```
In[40]:= ζ = Last[%]
```

```
Out[40]= -1 + i
```

Then all the cube roots of$2 + 2\,i$ are, according to the theory, the products of that one cube root and the cube roots of unity. And the cube roots of unity are just the powers of the primitive cube root of unity ω_3. So again the three cube roots of $2 + 2\,i$ are:

In[41]:=
```
ComplexExpand[ξ ω3^{0,1,2}]
```

Out[41]=
$$\left\{-1+i,\ \frac{1}{2}-\frac{\sqrt{3}}{2}+i\left(-\frac{1}{2}-\frac{\sqrt{3}}{2}\right),\ \frac{1}{2}+\frac{\sqrt{3}}{2}+i\left(-\frac{1}{2}+\frac{\sqrt{3}}{2}\right)\right\}$$

Just to be convinced that the theory really is correct, check that these numbers all do have cubes equal to$2 + 2\,i$:

In[42]:=
```
Simplify[%^3]
```

Out[42]=
```
{2 + 2 i, 2 + 2 i, 2 + 2 i}
```

Exploration: fourth, fifth, sixth, … roots of unity

Exercise. (a) Repeat for fourth roots of unity the calculations done above for cube roots of unity. How many do you obtain?
(b) Plot the fourth roots of unity as points in the plane and as the vertices of some polygon. What do you observe about this polygon?

Exercise. (a) Repeat for fifth roots of unity the calculations done above for cube roots of unity. How many do you obtain?
(b) Plot the fifth roots of unity as points in the plane and as the vertices of some polygon. What do you observe about this polygon?

Exercise. (a) Repeat for sixth roots of unity the calculations done above for cube roots of unity. How many do you obtain?
(b) Plot the sixth roots of unity as points in the plane and as the vertices of some polygon. What do you observe about this polygon?

Exercise. What does the evidence obtained so far suggest as to:
(a) how many *n*th roots of unity there are; and
(b) the configuration of these *n*th roots of unity as points in the complex plane?

Appendix: Plotting roots of unity as vectors from the origin

Instead of viewing the cube roots of unity just as points, you can also represent them by vectors—arrows drawn from the origin to those points. The graphics function **`ComplexArrow`** from *Presentations* draws such vectors. This function takes as argument a list of two complex numbers that prescribe the tail and the head of the arrow, respectively.

Consider, for example, the third cube root of unity (in the order that the three cube roots of unity were found)…

In[43]:=
```
cubeRootsUnity[[3]]
```

Out[43]= $-\frac{1}{2}+\frac{i\sqrt{3}}{2}$

…and form the arrow from the origin to that cube root of unity:

In[44]:=
```
ComplexArrow[{0, cubeRootsUnity[[3]]}]
```

Out[44]= $\text{Arrow}\left[\left\{\{0, 0\}, \left\{-\frac{1}{2}, \frac{\sqrt{3}}{2}\right\}\right\}\right]$

As you see, `ComplexArrow` creates an equivalent object with head `Arrow`. The function `Arrow` is a built-in *Mathematica* object. And what the Presentations `ComplexArrow` does is to apply `ToCoordinates` to each of the complex number arguments—here 0 and $-1/2 + i\sqrt{3}/2$—to convert them into the corresponding (x, y)-coordinate pairs `{0,0}` and `{-1/2,√2 /2}` required by `Arrow`. In other words,

```
ComplexArrow[{0,cubeRootsUnity[[3]]}]
```

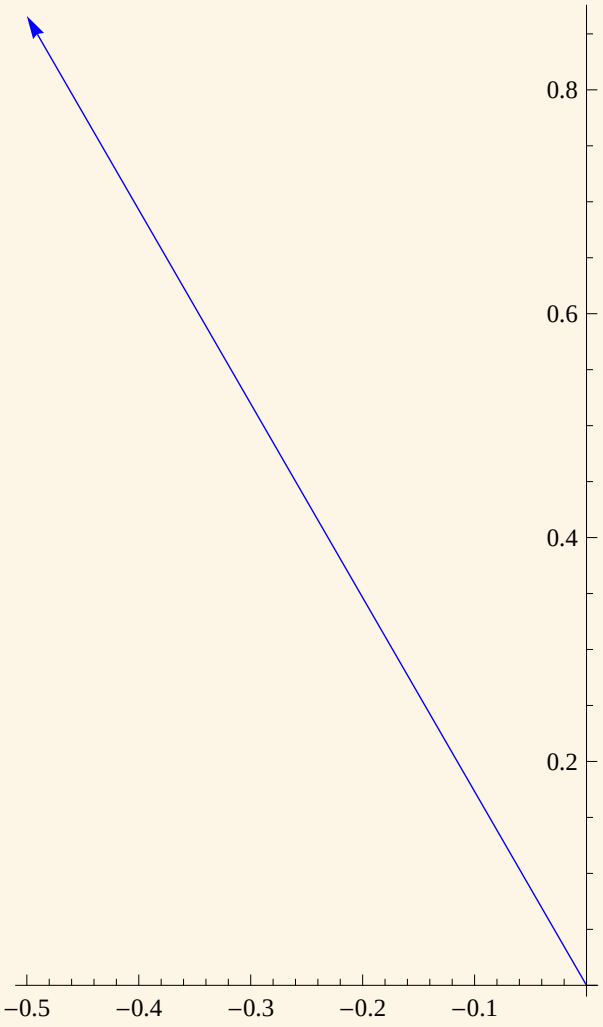
gives the same result as

```
Arrow[{ToCoordinates[0],ToCoordinates[cubeRootsUnity[[3]]]}]
```

Plot that vector:

In[45]:=
```
Draw2D[
 {
  Blue,
  ComplexArrow[{0, cubeRootsUnity[[3]]}]
 },
 Axes → True, ImageSize → 3 × 72]
```

Out[45]=



The following forms the list of vectors to all three cube roots of unity:

In[46]:=
```
Map[ComplexArrow[{0, #}] &, cubeRootsUnity]
```

Out[46]=
$$\left\{\text{Arrow}[\{\{0, 0\}, \{1, 0\}\}], \text{Arrow}\left[\left\{\{0, 0\}, \left\{-\frac{1}{2}, -\frac{\sqrt{3}}{2}\right\}\right\}\right], \text{Arrow}\left[\left\{\{0, 0\}, \left\{-\frac{1}{2}, \frac{\sqrt{3}}{2}\right\}\right\}\right]\right\}$$

To shorten such expressions, especially when they are part of larger expressions (such as `Draw2D` commands), you may use the special input form

func`/@`*list*,

to mean:

`Map[`*func*`,`*list*`]`

For example:

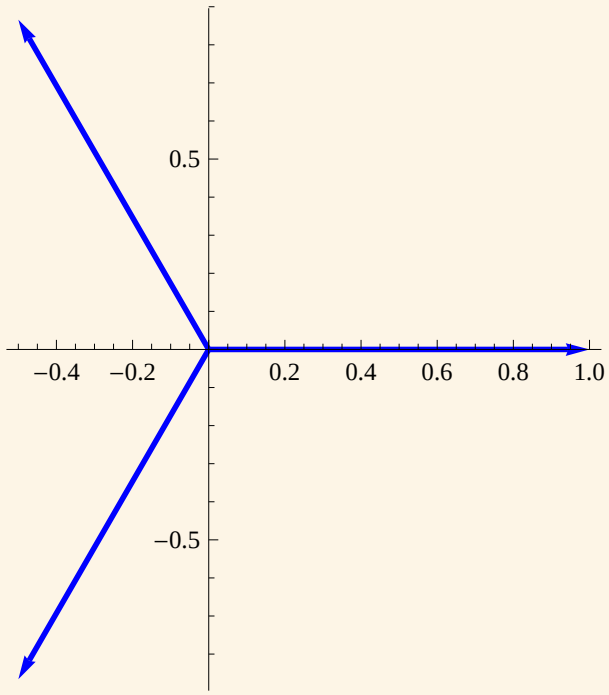
In[47]:=
```
ComplexArrow[{0, #}] & /@ cubeRootsUnity
```

Out[47]=
$$\left\{\text{Arrow}[\{\{0, 0\}, \{1, 0\}\}],\ \text{Arrow}\left[\left\{\{0, 0\}, \left\{-\frac{1}{2}, -\frac{\sqrt{3}}{2}\right\}\right\}\right], \text{Arrow}\left[\left\{\{0, 0\}, \left\{-\frac{1}{2}, \frac{\sqrt{3}}{2}\right\}\right\}\right]\right\}$$

And finally here's the desired plot of the vectors representing all three cube roots of unity:

In[48]:=
```
Draw2D[
 {
  Blue, Thick,
  ComplexArrow[{0, #}] & /@ cubeRootsUnity
 },
 Axes → True, ImageSize → 3 × 72]
```

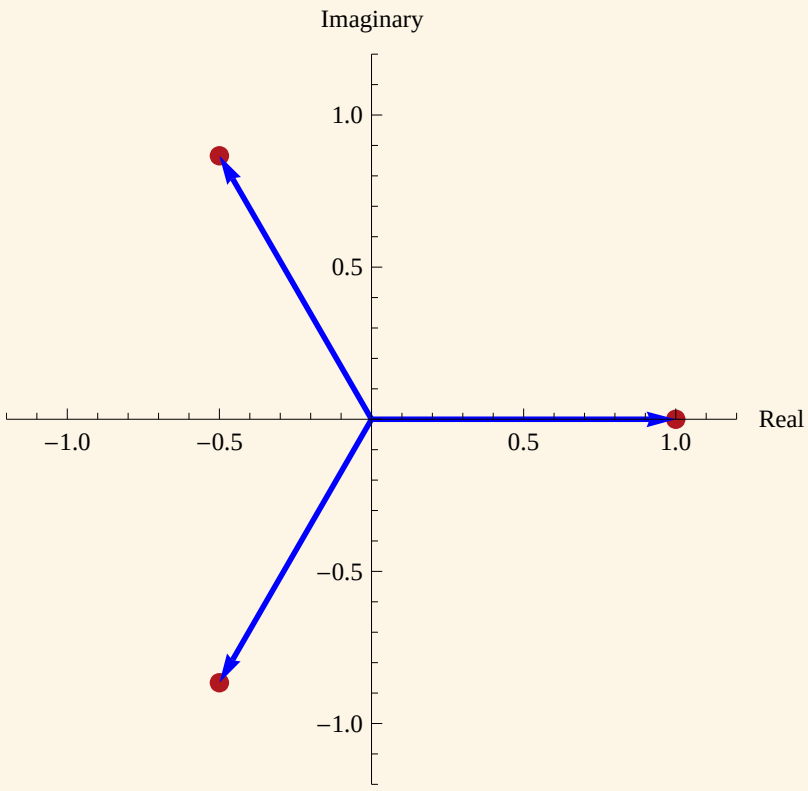
Out[48]=



It's easy to plot both the points and the vectors in the same graphics display:

In[49]:=
```
Draw2D[
 {
  Directive[Legacy@IndianRed, PointSize[Large]],
  ComplexPoint /@ cubeRootsUnity,
  Directive[Blue, Thick], ComplexArrow[{0, #}] & /@ cubeRootsUnity
 },
 PlotRange → {{-1.2, 1.2}, {-1.2, 1.2}},
 Axes → True, AxesLabel → {"Real", "Imaginary"}, ImageSize → 4 × 72]
```

Out[49]=



The `Directive` construction just used is a handy way of combining several graphics directives in order to make clear that they are going to be applied together to the graphics objects that follow (until some new directives override them).

In a similar way, you can display various graphics objects in a single `Draw2D` display.

Exercise. Include text labels for the three points in the preceding display .

Exercise. In a single `Draw2D` display, show the cube roots of unity, the triangle having them as vertices, and arrows going from the origin to these vertices.

Exercise. Repeat the preceding exercise but with the *sixth* roots of unity. (See the Exploration section.)